Apple® IIc Programmer's Guide to the 3.5 ROM

Apple® IIc Programmer's Guide to the 3.5 ROM

Apple® IIc Programmer's Guide to the 3.5 ROM

030-1196-A

Reorder Apple Product #A2L4037

Contents

Preface

Read Me First

This reference manual is intended for experienced Apple® IIc assembly-language programmers and hardware designers. It is a supplement to the *Apple IIc Reference Manual*. If you're a beginning user, you should learn about your Apple IIc and some of the programs, languages, and devices that you intend to use before you start reading this manual. A list of some Apple manuals that will help you learn about the operation and function of Apple II family computers is given at the end of this preface.

This manual assumes that you have access to Apple manuals describing the programs, languages, and devices that you intend to use with your Apple IIc.

About This Manual

The Programmer's Guide that you are reading describes the differences between the original Apple IIc and the Apple IIc with 32-kilobyte (32K) ROM.

3.5 ROM

The name *3.5 ROM* refers to the ROM that is described in this book as the *32K ROM*.

Chapter 1 details the changes brought about by the ROM enhancement. It also refers to other sources of information about Apple IIc operation.

Chapter 2 describes Monitor enhancements, including the Mini-Assembler and the STEP and TRACE routines.

Chapter 3 describes the use of the Protocol Converter and Protocol Converter calls.

Appendix A is an assembly listing of the firmware for the 32K ROM.

Where to Look for More Information

The following manuals have information about the function and programming of Apple II-family computers and some important peripherals.

These books are available from your Apple dealer:

- *About Your Enhanced Apple IIe: Programmer's Guide* (030-1143-A)
- *Apple IIc Owner's Manual* (030-1030-A)
- *Apple Pascal 1.2 Update Manual* (030-0602)
- *Apple Pascal 1.3 Update Manual* (030-1206-A)
- *Synertek Programming Manual* (A2L0003)

These books are available at your local bookstore:

- *Apple IIc Reference Manual* (Addison-Wesley 17727-7)
- *ProDos Technical Reference Manual* (Addison-Wesley 17728-5)

Watch for These

Look for these throughout the manual:

By the Way: Text set off in this manner presents sidelights or interesting pieces of information.

Important Text set off in this manner—with a tag in the margin—presents important information.

▲Warning Warnings like this indicate potential problems or disasters.

Apple IIe Text set off in this manner—with the name of an Apple computer or peripheral device in the margin—applies specifically to that computer or device.

Chapter 1 Overview

This chapter tells you about the operating differences between the original Apple® IIc and the Apple IIc with 32K ROM.

The following topics are discussed in this chapter:

- Physical changes in the Apple IIc
- Machine identification
- Features removed
- Interrupt handler revision
- Starting up from drives other than a Disk II
- New serial port commands
- Monitor enhancements, including the Mini-Assembler and the STEP and TRACE routines
- The Protocol Converter
- The Protocol Converter Bus

The Monitor enhancements and the Protocol Converter are described in detail in the following chapters.

Physical Changes

The 32K ROM for the Apple IIc replaces the original 16K ROM. The 32K ROM is organized into two 16K segments called the main and alternate banks. To toggle between banks, read address $C028. A write to $C028 toggles the line twice, leaving it unchanged.

Installing the new ROM involves cutting one trace and jumping another on the Apple IIc main logic board. Once the Apple IIc has been modified for the new ROM, the old ROM cannot be used in it.

▲Warning Improperly done modifications can damage your Apple IIc. Modifications performed by anyone other than an authorized Apple dealer void your warranty.

Machine Identification

Your Apple II program can tell which member of the Apple II family it is running on by checking the values in four locations of ROM (the *identification bytes*). Table 1-1 lists the machines and their identification bytes.

Table 1-1. Apple II Family Identification Bytes

Machine	$FBB3	$FB1E	$FBC0	$FBBF
Apple II	$38			
Apple II Plus	$EA	$AD		
Apple III (emulation)	$EA	$8A		
Apple IIe (original)	$06		$EA	
Apple IIe (enhanced)	$06		$E0	
Apple IIc (original)	$06		$00	$FF
Apple IIc (32K ROM)	$06		$00	$00

The only difference between the identification bytes for the original Apple IIc and the Apple IIc with 32K ROM is at $FBBF. That location was $FF in the original ROM and is $00 in the current revision of the 32K ROM.

If you're not sure whether a particular Apple IIc has the 32K ROM installed, follow the instructions in Chapter 2 for starting the Mini-Assembler. If you are successful (the Monitor prompt character changes from * to !), then the 32K ROM is present.

Features Removed

PR#7 (from BASIC) or 7 CONTROL-P (from the Monitor) no longer causes the system to **boot** from the external Disk II drive because that ROM space is now used for new features.

Interrupt Handler Revision

If the alternate bank of the ROM is being used, the interrupt handler switches the ROM to the main bank before calling a user's interrupt routine, and restores the alternate bank before returning from the interrupt. The bank of ROM to be returned to is indicated by the least significant bit of the byte stored in address $0044 after a break: 0 for the main bank, and 1 for the alternate bank of the ROM.

Programs written for the Apple II, II Plus, or IIe sometimes read $C02x before a BRK instruction that attempts to jump to a monitor routine. Reading $C02x toggles the cassette output signal in the Apple II, II Plus, and IIe, but switches between the main and alternate banks of ROM in the Apple IIc with 32K ROM. When the interrupt handler detects that a program has tried to jump to a monitor routine while in the alternate bank of the ROM, it automatically switches to the main bank of ROM and attempts to restart the program at the point before the break occurred. Note that this feature does not work for programs that try to read monitor data after a $C02x access because, in this case, no BRK instruction is executed.

Starting and Restarting

The Protocol Converter: see Chapter 3.

When the Apple IIc with 32K ROM is powered up, it attempts to boot from the built-in disk drive. If this fails, it attempts to boot from the first device attached to the Protocol Converter. If this fails too, the message `CHECK DISK DRIVE` is displayed and the system hangs.

The built-in Disk II can be booted with a `PR#6` command from BASIC, a [6] [CONTROL]-[P] from the Monitor, or `JMP $C600` from a machine-language program.

An external UniDisk™ 3.5 can be booted with a `PR#5` command from BASIC, a [5] [CONTROL]-[P] from the Monitor, or `JUMP $C500` from a machine-language program.

Important

External UniDisk 3.5 startup works with ProDOS®-based and Pascal 1.3 programs, but not with earlier versions of Pascal or with DOS.

New Serial Port Commands

The serial ports and the use of serial port commands are discussed in Chapters 7 and 8 of the *Apple IIc Reference Manual.*

Several new commands that can be used with serial ports 1 and 2 have been included in the 32K ROM. Each of these commands consists of two letters: the first letter identifies the command while the second letter enables (E) or disables (D) the function. You may separate the command and the letter with a space if you wish.

For example, to cause a line feed character to be output after every carriage return, use the command LE (or L E). To disable this function, use the command LD (or L D).

C D/E (Column overflow)	When enabled, this command causes a carriage return character to be sent automatically any time the column count exceeds the printer line width. Default = enabled.
F D/E (Find keyboard)	When enabled, this command causes your Apple IIc to accept signals coming from its keyboard as well as those coming over the serial port. You can use this command to disable the keyboard before receiving data or sending data to the printer, to prevent accidental keystrokes from disrupting the data flow. Be sure your program re-enables the keyboard when the data transfer is complete. This feature is available only in BASIC. Default = enabled.
L D/E (Line feed after carriage return)	When enabled, this command causes a line feed character to be output automatically after each carriage return character. LD and LE have the same effects as commands L (enable line feed) and K (disable line feed), which still work. Default = disabled.
M D/E (Mask line feeds)	When enabled, all incoming line feed characters are removed from the data stream. Default = enabled.
X D/E (XON/XOFF protocol)	When enabled, XON/XOFF protocol is used: the Apple IIc looks for any XOFF character (ASCII character DC3, decimal 19), and responds by halting transmission until an XON character (ASCII character DC1, decimal 17) is received. Default = disabled.

Monitor Enhancements

The Apple IIc with 32K ROM includes the Mini-Assembler, which lets you enter machine-language programs directly from the keyboard; and the STEP and TRACE Monitor routines, which facilitate debugging of machine-language programs. The Mini-Assembler and the STEP and TRACE routines are discussed in detail in Chapter 2.

Important Monitor commands can't be executed directly from the Mini-Assembler on an Apple IIc with 32K ROM.

The Protocol Converter

The Protocol Converter is a set of routines used to support I/O devices, such as the UniDisk 3.5, that connect to the external disk port. The routines begin at address $C500 in ROM. ProDOS and Pascal 1.3 recognize the Protocol Converter as a block device. The Protocol Converter and calls to the Protocol Converter are described in detail in Chapter 3.

The Protocol Converter Bus

The Protocol Converter Bus (CBus) consists of hardware and software components that permit and control communications between the Apple IIc and intelligent I/O devices (such as UniDisk 3.5's) connected to its external disk port.

The software part of the CBus includes the Protocol Converter and the CBus communication protocol.

The hardware component of the CBus is a daisy chain made up of the following:

- □ The Apple IIc disk port, using the disk controller unit (IWM), see Chapter 11 in the *Apple IIc Reference Manual.*
- □ One or more intelligent I/O devices (*bus residents*).
- □ One Disk IIc (optional). If included, the Disk IIc must be the terminal member of the daisy chain, and remains dormant when a bus resident is addressed.

The maximum number of bus residents is limited by the Apple IIc's power supply and IWM drive capacity. The software can support up to 127 bus residents.

Chapter 2 Monitor Enhancements

This chapter is about enhancements made to the Monitor, including the addition of the Mini-Assembler (which allows machine-language programs to be entered directly from the keyboard), and debugging routines for assembly-language programs. The following topics are discussed in this chapter:

- Procedures for using the Mini-Assembler
- The Mini-Assembler commands
- The STEP and TRACE debugging routines

The Mini-Assembler

Without an assembler, you have to write your machine-language program, take the hexadecimal values for the opcodes and operands, and store them in memory using the monitor commands described in Chapter 10 of the *Apple IIc Reference Manual.*

The Mini-Assembler lets you enter machine-language programs directly from the keyboard of your Apple. You can type ASCII characters in Mini-Assembler programs, exactly as you type them in the Monitor.

Important The Mini-Assembler doesn't accept labels; you must use actual hexadecimal values and addresses.

Starting the Mini-Assembler

To start the Mini-Assembler, first call the Monitor by typing `CALL-151` and pressing RETURN. Then from the Monitor, type `!` and press RETURN. The Monitor prompt character then changes from `*` to `!`.

When you finish using the Mini-Assembler, press RETURN from a blank line to return to the Monitor.

To enter code into memory, type the address, a colon, and the instruction. For example, after entering the Mini-Assembler, type

```
!300:STA C030
```

You can enter a series of instructions by typing a space, then the instruction, followed by RETURN:

```
!300:STA C030
! LDA #A0
! INX
```

Each succeeding instruction is placed in the next consecutive memory location. As you type in instructions, each is replaced by the starting address of the instruction, the hexadecimal value(s) of the instruction, followed by mnemonics describing the instruction. For example, the sequence of instructions given above produces the following on your display screen:

```
0300-   8D 30 C0        STA $C030
0303-   A9 A0           LDA #$A0
0305-   E8              INX
```

When you're ready to execute your program, press RETURN to leave the Mini-Assembler and return to the Monitor.

Important Monitor commands can't be executed directly from the Mini-Assembler on an Apple IIc with 32K ROM.

Using the Mini-Assembler

The Mini-Assembler saves one address, that of the program counter. Before you start to type a program, you must set the program counter to point to the location where you want the Mini-Assembler to store your program. Do this by typing the address followed by a colon.

Formats for operands are listed in Table 2-1.

After the colon, type the mnemonic for the first instruction in your program, followed by a space and the operand of the instruction. Now press RETURN. The Mini-Assembler converts the line you typed into hexadecimal, stores it in memory beginning at the location of the program counter, and then disassembles it again and displays the disassembled line. It then displays a prompt character on the next line.

Now the Mini-Assembler is ready to accept the second instruction in your program. To tell it that you want the next instruction to follow the first, don't type an address or a colon: just type a space and the next instruction's mnemonic and operand, then press RETURN. The Mini-Assembler assembles that line and waits for another.

If the line you type has an error in it, the Mini-Assembler causes the Apple IIc to beep and display a caret (^) under or near the offending character in the input line. Most common errors are the result of typographical mistakes: misspelled mnemonics, missing parentheses, and so forth. The Mini-Assembler also rejects the input line if you forget the space before or after a mnemonic or include an extraneous character in a hexadecimal value or address. If the destination address of a branch instruction is out of the range of the branch (more than 127 locations distant from the address of the instruction), the Mini-Assembler flags this as an error.

Dollar Signs: In this book, dollar signs ($) in addresses signify that the addresses are in hexadecimal notation. The dollar signs are ignored by the Mini-Assembler and may be omitted when typing programs.

```
!300:LDX #02

0300-    A2 02        LDX     #$02
! LDA $0,X

0302-    B5 00        LDA     $00,X
! STA $10,X

0304     95 10        STA     $10,X
! DEX

0306-    CA           DEX
! STA $C030

0307-    8D 30 C0     STA     $C030
! BPL $302

030A-    10 F6        BPL     $0302
! BRK

030C-    00           BRK
!
```

To leave the Mini-Assembler and return to the Monitor, press RETURN at a blank line.

Your assembly-language program is now stored in memory. You can display it with the LIST command:

```
*300L

0300-   A2 02         LDX    #$02
0302-   B5 00         LDA    $00,X
0304-   95 10         STA    $10,X
0306-   CA            DEX
0307-   8D 30 C0      STA    $C030
030A-   10 F6         BPL    $0302
030C-   00            BRK
030D-   00            BRK
030E-   00            BRK
030F-   00            BRK
0310-   00            BRK
0311-   00            BRK
0312-   00            BRK
0313-   00            BRK
0314-   00            BRK
0315-   00            BRK
0316-   00            BRK
0317-   00            BRK
0318-   00            BRK
0319-   00            BRK
*
```

Mini-Assembler Instruction Formats

The Apple IIc Mini-Assembler recognizes 66 mnemonics and 15 addressing formats. The mnemonics are standard, as used in the *Synertek Programming Manual*, but the addressing formats are somewhat different; see Table 2-1.

Table 2-1. Mini-Assembler Address Formats

Addressing Mode	Format
Accumulator	*
Implied	*
Immediate	#${value}
Absolute	${address}
Zero page	${address}
Indexed zero page	${address},X ${address},Y
Indexed absolute	${address},X ${address},Y
Relative	${address}
Indexed indirect	(${address},X)
Indirect indexed	(${address}),Y
Absolute indirect	(${address})

* These instructions have no operands.

An address consists of one or more hexadecimal digits. The Mini-Assembler interprets addresses the same way the Monitor does: if an address has fewer than four digits, the Mini-Assembler adds leading zeros; if the address has more than four digits, then it uses only the last four.

There is no syntactical distinction between the absolute and zero page addressing modes. If you give an instruction to the Mini-Assembler that can be used in both absolute and zero page mode, the Mini-Assembler assembles that instruction in absolute mode if the operand for that instruction is greater than $FF, and it assembles it in zero page mode if the operand is less than $100.

Instructions in accumulator mode and implied addressing mode need no operands.

Branch instructions, which use the relative addressing mode, require the target address of the branch. The Mini-Assembler automatically calculates the relative distance to use in the instruction. If the target address is more than 127 locations distant from the instruction, the Mini-Assembler sounds a bell (beep), displays a caret (^) under the target address, and does not assemble the line.

If you give the Mini-Assembler the mnemonic for an instruction and an operand, and the addressing mode of the operand cannot be used with the instruction you entered, then the Mini-Assembler will not accept the line.

STEP and TRACE

STEP and TRACE are Monitor facilities for debugging assembly-language programs. The STEP command decodes, displays, and executes one instruction at a time, and the TRACE command steps continuously through a program, stopping when a BRK instruction is executed or [] is pressed. You can press [] to slow down the trace to one step per second.

Each STEP command causes the Monitor to execute the instruction in memory pointed to by the Program Counter. The instruction is displayed in its disassembled form, then executed. The contents of the 65C02's internal registers are displayed after the instruction is executed. After execution, the Program Counter is incremented to point to the next instruction in the program.

Here is an example of the STEP command, using the following program:

```
$0300:     LDX #02
$0302:     LDA $00,X
$0304:     STA $10,X
$0306:     DEX
$0307:     STA $C030
$030A:     BPL $0302
$030C:     BRK
```

To step through this program, first call the Monitor by typing `CALL-151` and pressing [RETURN], and then from the Monitor, type **`300S`** (to start the STEP routine at address $0300). Type [S] to advance each additional step through the program. The Monitor keeps the Program Counter and the last opened address separate from one another, so you can examine or change the contents of memory while you are stepping through your program.

Here's what happens when you step through the program above, examining the contents of location $0012 after the third step. Note that in this example, what you type appears just after the * prompt, and the information on the next two lines—that begin without the * prompt—is what the computer displays on the screen in response.

```
*300S

0300-   A2 02     LDX #02
M=CA A=0A X=02 Y=D8 P=30 S=F8
*S

0302-   B5 00     LDA $00,X
M=CA A=0C X=02 Y=D8 P=30 S=F8
*S

0304-   95 10     STA $10,X
M=CA A=0C X=02 Y=D8 P=30 S=F8
*12

0012- 0C
*S

0306-   CA        DEX
M=CA A=0C X=01 Y=D8 P=30 S=F8
*S

0307-   8D 30 C0 STA $C030
M=CA A=0C X=01 Y=D8 P=30 S=F8
*S

030A-   10 F6     BPL $0302
M=CA A=0C X=01 Y=D8 P=30 S=F8
*S

0302-   B5 00     LDA $00,X
M=CA A=0B X=01 Y=D8 P=30 S=F8
*S

0304-   95 10     STA $10,X
M=CA A=0B X=01 Y=D8 P=30 S=F8
*
```

The TRACE command is a continuous version of the STEP command; it will stop stepping through the program only when you press , or when it encounters a BRK instruction in the program. Press to slow the trace to one step per second.

Cautions

Keep the following cautions in mind when using the STEP and TRACE Monitor commands:

- If the program ends with an RTS instruction, the TRACE routine will continue to run indefinitely until stopped with .
- You can't step or trace through routines that use the same zero page locations as the Monitor.

ASCII Input Mode

This mode lets you enter ASCII characters as well as their hexadecimal ASCII equivalents. This means that 'A is the same as C1 and 'B is the same as C2 to the Monitor. The ASCII value for *any* character following an apostrophe is used by the Monitor. For example, to enter the string "Hooray for sushi!" at $300 in memory, type

```
*3ØØ:'H 'o 'o 'r 'a 'y ' 'f 'o 'r ' 's 'u 's 'h 'i '!
```

Note that each character to be placed in memory is delimited by a leading ' and a trailing space. The only exception to this rule is that the last character in the line is followed by a RETURN character instead of a space.